Can Soft Goals be Compiled Away More Efficiently?

Alberto Pozanco

J.P. Morgan AI Research
alberto.pozancolancho@jpmorgan.com

Abstract

In many scenarios, planning agents do not have enough resources to achieve all of their goals. In those cases, net-benefit planning aims to maximize the utility of the goals achieved by the plan, potentially ignoring some goals if the cost of achieving them is too high. Keyder and Geffner (2009) showed that these tasks can be reformulated into classical planning. In this paper we slightly modify their compilation to reduce the state and plan spaces by allowing the planner to finish only in those states increasing utility, i.e., when a soft goal has been achieved. Experimental results show that, although the new reformulation yields smaller state and plan spaces, the resulting tasks turn out to be more challenging. We conjecture this is due to the uninformative heuristic regions generated by the new state space, and call for further research on this topic.

Introduction

Classical planning is the task of choosing and organizing a sequence of deterministic actions such that, when applied in a given initial state, it results in a state in which all goals are true (Ghallab, Nau, and Traverso 2004). This setting does not fit many real-world planning scenarios, where agents often do not have enough resources to achieve all of their goals. In those cases, partial satisfaction planning aims to maximize the utility of the goals achieved by the plan, potentially ignoring some goals if the cost of achieving them is too high. There exist two main partial satisfaction planning frameworks in the literature: (i) *oversubscription planning* (Smith 2004), where action costs and goal utilities are not comparable, and the objective is to maximize the utility of the goals subject to a cost budget (bound); and *net-benefit planning* (Van Den Briel et al. 2004), where action costs and goal utilities are assumed to be comparable, and the quality of a plan is measured as the difference between the utility of the obtained end state and the solution cost.

In this paper we focus on the latter setting, and more specifically in Keyder and Geffner (2009) reformulation of net-benefit planning tasks into classical planning. This compilation augments the original task with extra actions that encode the decision to achieve the goal without increasing the plan's cost, or ignore it by paying a penalty. This transformation opens the door to solving net-benefit planning tasks with any existing planner, and has been widely used by many planning works and applications (Cenamor et al. 2017; Torralba et al. 2021; Pozanco, Fernández, and Borrajo 2021; Klassen et al. 2022). Keyder and Geffner's compilation allows the planner to ignore the remaining goals and finish at any state during the search. This task structure has two main drawbacks. First, it entails larger state spaces, which can ultimately lead to more challenging tasks. Second, it generates a large number of valid plans with very low quality, which is usually a problem when using greedy algorithms to solve the reformulated tasks.

In this paper we slightly modify Keyder and Geffner (2009) compilation to tackle these two problems, i.e., reducing the state space and the number of valid plans. This is done by allowing the planner to finish the search only in those states increasing utility, i.e., when a soft goal has been achieved. We empirically evaluate this new compilation in a set of net-benefit planning tasks (Katz et al. 2019). Preliminary results show that, although the new reformulation yields smaller state and plan spaces, the resulting tasks turn out to be more challenging for most search algorithms and heuristics, i.e., they need to expand and generate more nodes. We conjecture this is due to the uninformative heuristic regions (Xie, Müller, and Holte 2014) generated by the new state space, and call for further research on this topic.

Background

Classical Planning

A STRIPS classical planning task can be defined as follows:

Definition 1 *A* STRIPS ***planning task*** *can be defined as a tuple $\mathcal{P} = \langle F, A, I, G, c\rangle$, where F is a set of propositions, A is a set of instantiated actions, $I \subseteq F$ is an initial state, $G \subseteq F$ is a goal state, and $c : A \mapsto \mathbb{R}_0^+$ is the cost associated to each action.*

A state consists of a set of propositions $s \subseteq F$ that are true at a given time. A state is totally specified if it assigns truth values to all the propositions $p \in F$, as the initial state I of a planning task. A state is partially specified (partial state) if it assigns truth values to only a subset of the propositions in F, as the conjunction of propositions G of a planning task. Each action $a \in A$ is described by a set of preconditions (pre(a)), which represent literals that must be true in a state to execute an action, and a set of effects (eff(a)), which are the literals that are added (add(a) effects) or removed (del(a) effects) from the state after the action execution. We write actions as tuples of the form $a = \langle \text{pre}(a), \text{eff}(a) \rangle$.

The execution of an action a in a state s is defined by a function γ such that $\gamma(s, a) = (s \setminus \text{del}(a)) \cup \text{add}(a)$ if $\text{pre}(a) \subseteq s$, and s otherwise (it cannot be applied). The output of a planning task is a sequence of actions, called a plan, $\pi = (a_1, \ldots, a_n)$. The execution of a plan π in a state s can be defined as:

$$\Gamma(s, \pi) = \begin{cases} \Gamma(\gamma(s, a_1), (a_2, \ldots, a_n)) & \text{if } \pi \neq \emptyset \\ s & \text{if } \pi = \emptyset \end{cases}$$

A plan π is valid if $G \subseteq \Gamma(I, \pi)$. The plan cost is commonly defined as

$$c(\pi) = \sum_{a_i \in \pi} c(a_i). \tag{1}$$

A plan with minimal cost is called optimal.

Net-benefit Planning

While in classical planning a plan is only valid if all goals (propositions) are achieved, *net-benefit planning* (Van Den Briel et al. 2004) relaxes this assumption and allows plans that do not fully satisfy all the goals but maximize a utility function. Formally:

Definition 2 *A* STRIPS ***net-benefit planning task*** *can be defined as a tuple $\mathcal{P}_u = \langle \mathcal{P}, u \rangle$ where $\mathcal{P}$ is a planning task and u is a partial function $u : F \mapsto \mathbb{R}^+$ that maps a subset of propositions (the soft goals) into positive reals.*

In a STRIPS net-benefit planning task $\mathcal{P}_u$, the *utility* of a plan is given by the difference between the total utility obtained by the plan and its cost:

$$u(\pi) = \sum_{p \in \Gamma(I, \pi)} u(p) - c(\pi) \tag{2}$$

A plan π optimally solves a net-benefit planning task when no other plan π has a utility $u(\pi')$ higher than u. The implicit assumption of $u(\pi)$ is that action costs and goal utilities are comparable, and solution quality is measured as the difference between the utility of the obtained end state and the solution cost.

Soft Goals Can be Compiled Away

Keyder and Geffner (2009) proposed the following compilation of the net-benefit planning task into a standard planning task. We will refer to it as K:

Definition 3 *Given a* STRIPS *net-benefit planning task $\mathcal{P}_u$, a K* STRIPS ***planning task*** *is $\mathcal{P}_K = \langle F_K, A_K, I_K, G_K, c_K \rangle$ with*

- $F_K = F \cup S'(\mathcal{P}) \cup \bar{S}(\mathcal{P}) \cup \{\textit{normal-mode}, \textit{end-mode}\}$
- $A_K = A' \cup \{\textit{collect}(p), \textit{forgo}(p) \mid p \in SG(\mathcal{P})\} \cup \{\textit{end}\}$
- $I_K = I \cup \bar{S}(\mathcal{P}) \cup \{\textit{normal-mode}\}$
- $G_K = G \cup S'(\mathcal{P})$
- $c_K(a) = \begin{cases} c(a) & \textit{if } a \in A' \\ u(p) & \textit{if } a = \textit{forgo}(p) \\ 0 & \textit{if } a = \textit{collect}(p) \textit{ or } a = \textit{end} \end{cases}$

where

- $SG(\mathcal{P}) = \{p \mid (p \in F) \wedge (u(p) > 0)\}$
- $S'(\mathcal{P}) = \{p' \mid p \in SG(\mathcal{P})\}$
- $\bar{S}(\mathcal{P}) = \{\bar{p'} \mid p' \in S'(\mathcal{P})\}$
- $\textit{end} = \langle \{\textit{normal-mode}\}, \{\textit{end-mode}, \neg\textit{normal-mode}\} \rangle$
- $\textit{collect}(p) = \langle \{\textit{end-mode}, p, \bar{p'}\}, \{p', \neg\bar{p'}\} \rangle$
- $\textit{forgo}(p) = \langle \{\textit{end-mode}, \bar{p}, \bar{p'}\}, \{p', \neg\bar{p'}\} \rangle$
- $A' = \{\langle \textit{pre}(a) \cup \{\textit{normal-mode}\}, \textit{eff}(a) \rangle \mid a \in A\}$

For each soft goal p in $\mathcal{P}_u$, the compilation adds a hard goal p' in $\mathcal{P}_K$ that can be achieved in two ways: with the *collect*(p) action, that has cost 0 but requires p to be true; or with the *forgo*(p) action, that has cost equal to the utility of p yet can be performed when p is false, or equivalently when $\bar{p}$ is true. These two actions can be executed only after the *end* action makes the *end-mode* proposition true, while the actions from the original problem $\mathcal{P}$ can be executed only when the *normal-mode* proposition is true, i.e., before executing the *end* action. Moreover, each soft goal p can only be achieved by either a *collect* or *forgo* action, as both delete their shared precondition $\bar{p'}$, which no action makes true. As there is no way to make *normal-mode* true again after it is deleted by the *end* action, all plans π_K that solve the reformulated task $\mathcal{P}_K$ have the form $\pi_K = \langle \pi, \text{end}, \pi' \rangle$, where π is a plan for $\mathcal{P}$ and π' is a sequence of $|S'(\mathcal{P})|$ *collect* and *forgo* actions in any order, the former appearing when $p \in \Gamma(I, \pi)$, and the latter otherwise. As proven in (Keyder and Geffner 2009), finding an optimal (maximum utility) plan for $\mathcal{P}_u$ is equivalent to finding an optimal (minimum cost) plan for the reformulated task $\mathcal{P}_K$.

Pruning the State Space

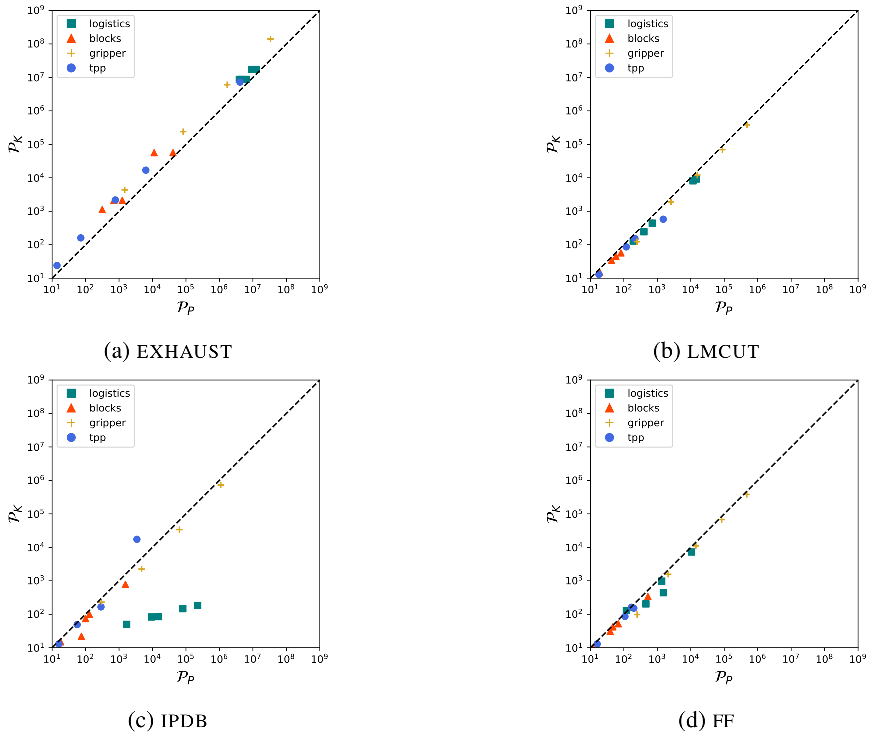
Keyder and Geffner's compilation allows the planner to ignore the remaining goals and finish at any state during the search. This generates many states and valid plans that can be pruned by slightly modifying the compilation to allow finishing planning only in those states increasing utility. We will refer to this pruned compilation as P, and formally define it as follows:

Definition 4 *Given a* STRIPS *net-benefit planning task $\mathcal{P}_u$, a P* STRIPS ***planning task*** *is $\mathcal{P}_P = \langle F_P, A_P, I_P, G_P, c_P \rangle$ with*

- $F_P = F_K \cup \{\textit{endable}\}$
- $A_P = A_e \cup A_{\neg e} \cup \{\textit{collect}(p), \textit{forgo}(p) \mid p \in SG(\mathcal{P})\} \cup \{\textit{end, continue}\}$
- $I_K = I \cup \bar{S}(\mathcal{P}) \cup \{\textit{endable}\}$
- $G_K = G \cup S'(\mathcal{P})$
- $c_P = c_K \cup \{0 \textit{ if } a = \textit{continue}\}$

where everything remains as in $\mathcal{P}_K$ except for

- $\textit{end} = \langle \{\textit{endable}\}, \{\textit{end-mode}, \neg\textit{normal-mode}, \neg\textit{endable}\} \rangle$
- $\textit{continue} = \langle \{\textit{endable}\}, \{\textit{normal-mode}, \neg\textit{endable}\} \rangle$
- $A_e = \{\langle \textit{pre}(a) \cup \{\textit{normal-mode}, \neg\textit{endable}\}, \textit{eff}(a) \cup \{\neg\textit{normal-mode}, \textit{endable}\} \rangle \mid (a \in A \mid G \cap \textit{eff}(a) \neq \emptyset)\}$
- $A_{\neg e} = \{\langle \textit{pre}(a) \cup \{\textit{normal-mode}, \neg\textit{endable}\}, \textit{eff}(a) \rangle \mid (a \in A \mid G \cap \textit{eff}(a) = \emptyset)\}$



(a) EXHAUST

(b) LMCUT

(c) IPDB

(d) FF

Figure 1: Pairwise comparison of the number of expanded nodes in log scale. Each point in the plots corresponds to a problem, with its color indicating the domain it belongs to. Points above the diagonal indicate that the given configuration needs to expand less states when solving $\mathcal{P}_P$ than when solving $\mathcal{P}_K$.

In this new compilation, we introduce a new *endable* proposition that serves as a flag indicating when we can switch from *normal-mode* to *end-mode*. This proposition is true in the initial state, as in some settings that could be the state with the higher utility. When *endable* is true, only two actions are applicable: (i) *end*, which will finish planning and start the sequence of *collect* and *forgo* actions; and (ii) *continue*, which makes *endable* false and activates *normal-mode*, indicating that the actions from the original problem can be executed. The *endable* proposition is only made true again when the plan achieves a soft goal, i.e., after applying one of the actions in A_e. Therefore, it is only in these states when we can switch to *end-mode*, as opposed to $\mathcal{P}_K$, which allows the planner to switch to *end-mode* from any state.

All plans π_P that solve the reformulated task $\mathcal{P}_P$ have the same form as those solving $\mathcal{P}_K$, with the only exception that they will have a number of *continue* actions interleaved in π, depending on how many soft goals π achieves. Following the equivalence proof by (Keyder and Geffner 2009) we can easily prove that since the only extra action that we include has cost 0, finding an optimal (maximum utility) plan for $\mathcal{P}_u$ is equivalent to finding an optimal (minimum cost) plan for the reformulated task $\mathcal{P}_P$.

Evaluation

We compare both reformulations in a set of planning tasks taken from the PDDL benchmarks for oversubscription planning (Katz et al. 2019)[1]. We remove the cost bound so we can turn the oversubscription tasks into net-benefit planning tasks. In order to conduct a preliminary evaluation, we selected the first 5 problems for each of the following 4 domains: BLOCKS-WORLD, GRIPPER, LOGISTICS, and TPP. This gives us 20 different planning tasks that we solve with A* (Hart, Nilsson, and Raphael 1968) and 3 different heuristics, as they are implemented in Fast Downward (Helmert 2006): LMCUT (Helmert and Domshlak 2009), IPDB (Haslum et al. 2007), and FF (Hoffmann and Nebel 2001), which is the only inadmissible heuristic in the set. We also run a version of Fast Downward that expands all the reachable states, so we can analyze the state spaces generated by both compilations. We refer to this algorithm as EXHAUST. We measure the number of nodes expanded by each configuration across the 20 planning tasks. Experiments were run on an Apple M1 with 8GB of memory and a timeout of 600 seconds.

Measuring the Resulting State Spaces

Figure 1 shows the results of our evaluation as a pairwise comparison of the number of expanded nodes when solving $\mathcal{P}_K$ (y axis) and $\mathcal{P}_P$ (x axis) with the different configurations. As we can see in Figure 1a, where we run both reformulations with EXHAUST, the state spaces generated by our new compilation are always smaller than those induced by

[1]https://zenodo.org/records/2576024

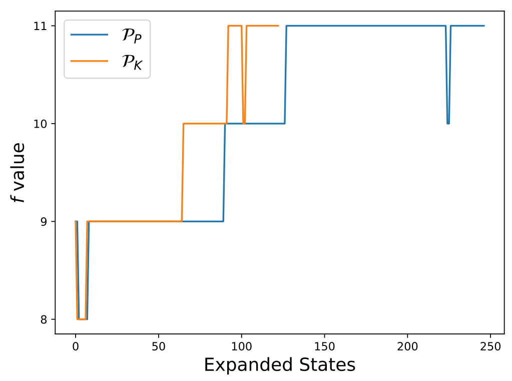


Figure 2: f value (y axis) of the node selected to be expanded at each iteration of A* (x axis) when solving the smallest GRIPPER instance with the LMCUT heuristic. The orange line shows the algorithm solving $\mathcal{P}_K$, and the blue line the algorithm solving $\mathcal{P}_P$.

Keyder and Geffner's compilation. In some BLOCKS problems, $\mathcal{P}_P$ achieves a reduction of up to 80% expanded states, i.e., our compiled tasks have 20% of the states of $\mathcal{P}_K$. In the largest GRIPPER problem, $\mathcal{P}_P$ has $\approx$ 33M (million) states versus the $\approx$ 140M states of $\mathcal{P}_K$, therefore representing a saving of $\approx$ 106M states. This behavior is consistent across domains and problems, indicating that our compilation is achieving its purpose of pruning the state space.

Planning in the Reformulated Tasks

Next, we wanted to evaluate whether the new compilation helps planning, i.e., if solving $\mathcal{P}_P$ is faster than solving $\mathcal{P}_K$. Figures 1b, 1c and 1d show the number of nodes expanded by LMCUT, IPDB and FF, respectively. As we can see, $\mathcal{P}_P$ is not amenable for the three configurations: although its state space is smaller than that of $\mathcal{P}_K$, the algorithms need to expand more states to find a solution. For example, LMCUT needs to expand up to 2.6 times more states to find an optimal solution in $\mathcal{P}_P$ than in $\mathcal{P}_K$. This difference is even worse when solving the reformulated tasks with IPDB. In a LOGISTICS task, IPDB finds a solution to $\mathcal{P}_K$ in less than a second, expanding only 184 states, while it needs 10 seconds and 223 335 expanded states in $\mathcal{P}_P$.

We conjecture this bad performance is due to the uninformative heuristic regions (Xie, Müller, and Holte 2014) generated by the new state space. In the new reformulation, we are adding *continue*, a zero cost action that decides whether to finish planning or keep achieving soft goals. Reasoning about this decision, as well as introducing a zero cost action, seems to negatively affect the search performance. In order to shed some light into this, we analyzed the search behavior by getting the f value of the node selected to be expanded at each iteration of A*. Figure 2 shows the results of this analysis when solving the smallest GRIPPER instance with the LMCUT heuristic. As we can see, when solving $\mathcal{P}_P$, A* encounters some f value plateaus, with more states having an f value of 9 and 10 compared to those encountered when solving $\mathcal{P}_K$.

Conclusions and Future Work

In this paper we slightly modified Keyder and Geffner's soft goals compilation to reduce the generated state space. This is done by allowing the planner to finish only in those states increasing utility, i.e., when a soft goal has been achieved. Preliminary results in some planning instances show that, although the new reformulation yields smaller state spaces, the resulting tasks turn out to be more challenging for most search algorithms. Therefore, one should still use Keyder and Geffner's (2009) compilation over ours when facing net-benefit planning tasks.

This result opens the door to some interesting research questions.

1. Further study $\mathcal{P}_P$. In our preliminary evaluation, we only tested one search algorithm (A*) and few heuristics in a limited number of domains and problems. We would like to broaden the scope of the evaluation to be able to draw more solid conclusions. Moreover, we would like to investigate potential patterns: how does the number of soft goals affect the search behavior? Are there any combinations of search algorithm and heuristic that benefit from the new compilation? Are there any domains where $\mathcal{P}_P$ generates easier tasks than $\mathcal{P}_K$?
2. Can $\mathcal{P}_P$ be improved? We would like to analyze the full search tree in small instances to see how heuristics assign values to each state. This would allow us to better understand why the compilation generates heuristic plateaus, and potentially provide us useful insights on how to improve it.
3. Adversarial attacks to planning domains. By slightly modifying a planning task (Keyder and Geffner's compilation), we have generated a new planning task that is much harder (sometimes even orders of magnitude) than the original one. Many interesting research questions arise from this result. Can planning domains (or planning tasks) be minimally modified (or *attacked*) to disrupt the behavior of search algorithms? What is the definition of minimal changes? Can we build robust domains (or planning tasks) that are more difficult to be attacked?

We hope this negative result and some of the above research questions serve as a source of discussion and inspiration for the planning community.

Acknowledgements

This paper was prepared for informational purposes by the Artificial Intelligence Research group of JPMorgan Chase & Co. and its affiliates ("JP Morgan"), and is not a product of the Research Department of JP Morgan. JP Morgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a

solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful.

References

Cenamor, I.; de la Rosa, T.; Núñez, S.; and Borrajo, D. 2017. Planning for tourism routes using social networks. *Expert Systems with Applications*, 69: 1–9.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2004. *Automated planning - theory and practice*. Elsevier. ISBN 978-1-55860-856-6.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cybern.*, 4(2): 100–107.

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; and Koenig, S. 2007. Domain-Independent Construction of Pattern Database Heuristics for Cost-Optimal Planning. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, 1007–1012. AAAI Press.

Helmert, M. 2006. The Fast Downward Planning System. *J. Artif. Intell. Res.*, 26: 191–246.

Helmert, M.; and Domshlak, C. 2009. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway? In Gerevini, A.; Howe, A. E.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009, Thessaloniki, Greece, September 19-23, 2009*. AAAI.

Hoffmann, J.; and Nebel, B. 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. *J. Artif. Intell. Res.*, 14: 253–302.

Katz, M.; Keyder, E.; Winterer, D.; and Pommerening, F. 2019. Oversubscription planning as classical planning with multiple cost functions. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 29, 237–245.

Keyder, E.; and Geffner, H. 2009. Soft goals can be compiled away. *Journal of Artificial Intelligence Research*, 36: 547–556.

Klassen, T. Q.; McIlraith, S. A.; Muise, C.; and Xu, J. 2022. Planning to avoid side effects. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 9830–9839.

Pozanco, A.; Fernández, S.; and Borrajo, D. 2021. On-line modelling and planning for urban traffic control. *Expert Systems*, 38(5): e12693.

Smith, D. E. 2004. Choosing objectives in over-subscription planning. In *Proceedings of the Fourteenth International Conference on International Conference on Automated Planning and Scheduling*, 393–401.

Torralba, Á.; Speicher, P.; Künnemann, R.; Steinmetz, M.; and Hoffmann, J. 2021. Faster stackelberg planning via symbolic search and information sharing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 11998–12006.

Van Den Briel, M.; Sanchez, R.; Do, M. B.; and Kambhampati, S. 2004. Effective approaches for partial satisfaction (over-subscription) planning. In *Proceedings-Nineteenth National Conference on Artificial Intelligence (AAAI-2004)*, 562–569.

Xie, F.; Müller, M.; and Holte, R. 2014. Adding local exploration to greedy best-first search in satisficing planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.